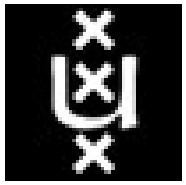


Applied Machine Learning
Linear Regression

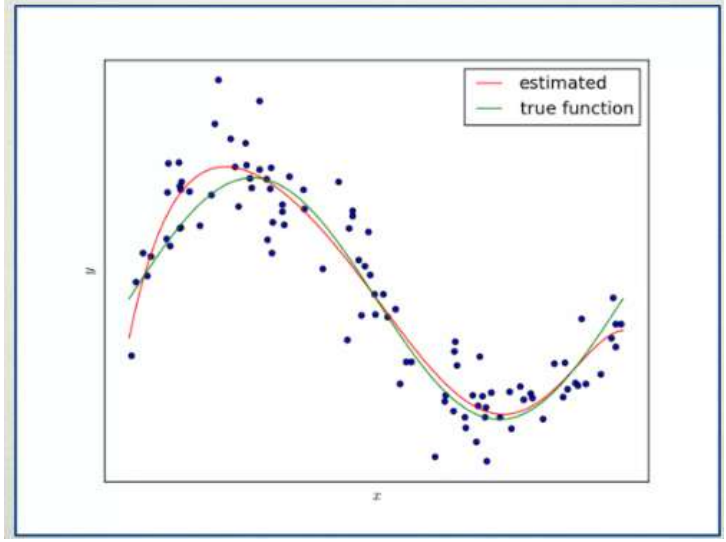
BSc course Informatiekunde 2026
https://staff.fnwi.uva.nl/a.visser/education/AML

Arnoud Visser
Intelligent Robotics Lab & Computer Vision Lab
Informatics Institute
Universiteit van Amsterdam
A.Visser@uva.nl

Illustrations courtesy of Maarten Marx, Sarah Guido, Yolanda Hagar, and many others.

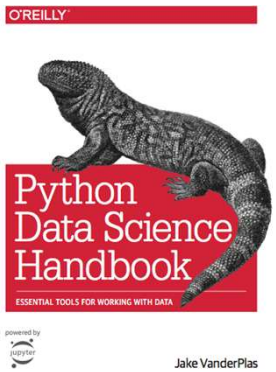
Machine Learning & Data Science

"Machine learning is where the computational and algorithm skills of data science meet the statistical thinking of data science."



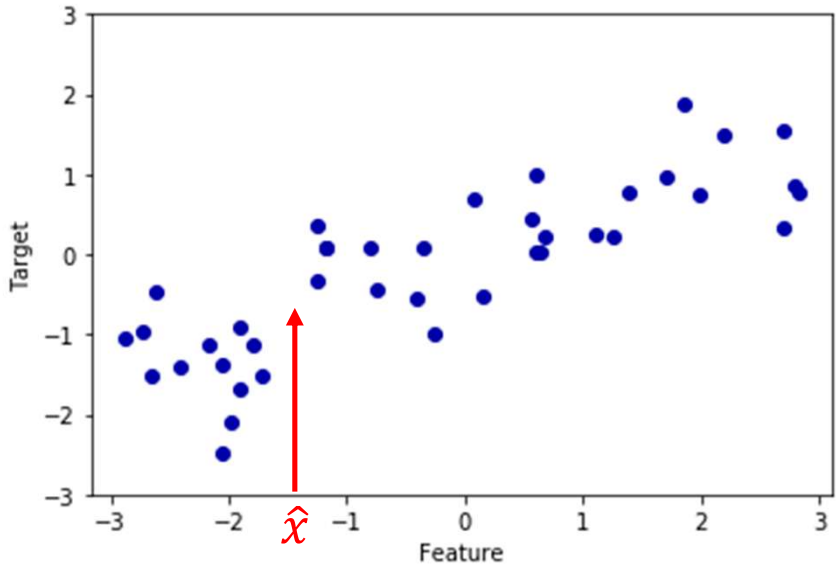
- Given training data predict the values in between
  – the function between input and output

Jake VanderPlas, Python Data Science Handbook,
O'Reilly Media, November 2016



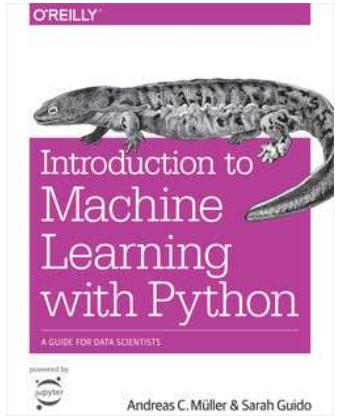
Introduction to Machine Learning

Predict for this dataset the value at $\hat{x} = -1.5$.



- We need the function between input X and output y

Between Feature X and Target y



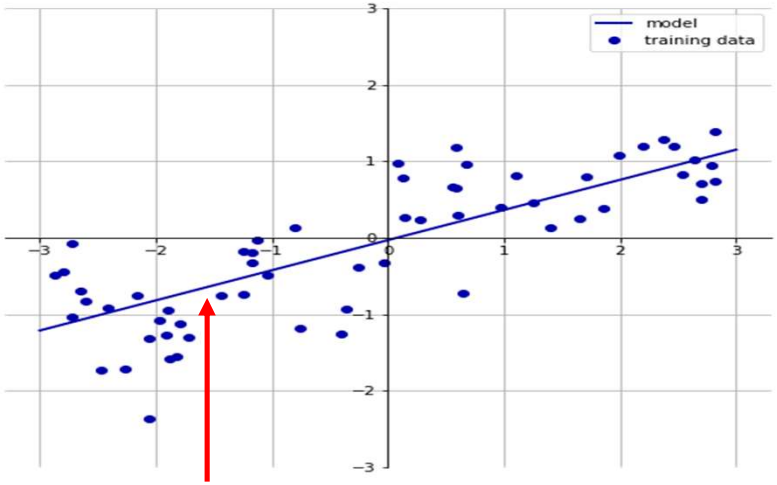
Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Figure 2-3

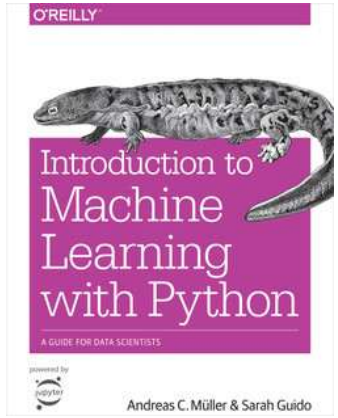
Introduction to Machine Learning

We need the function between input X and output y

.



- the function $y = f(X)$ the model



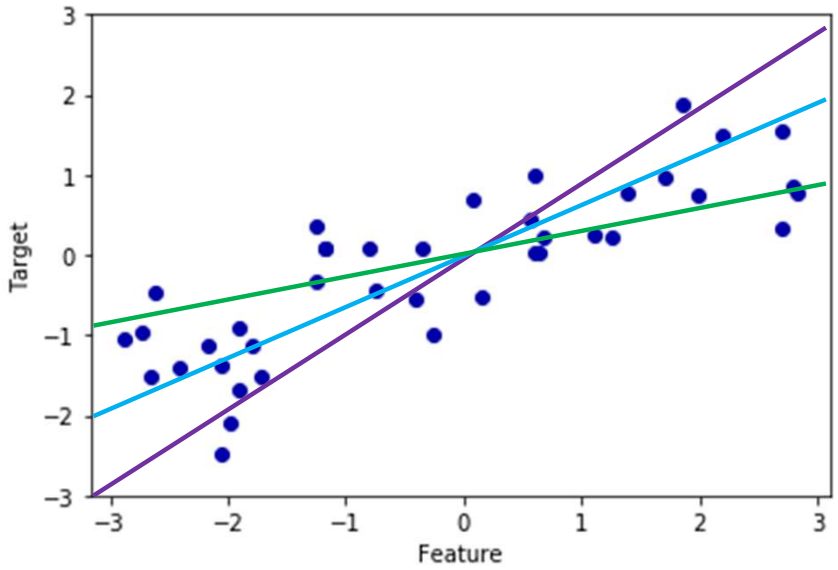
Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Figure 2-11

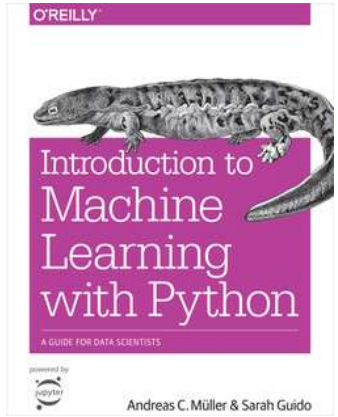
Introduction to Machine Learning

We fit a line through the points *y*

.



- Eyeball method the model

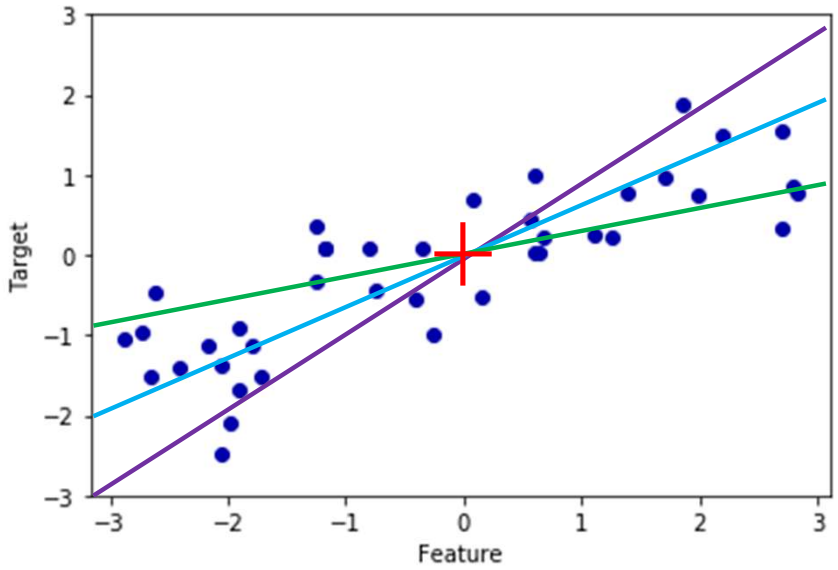


Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Figure 2-3

Introduction to Machine Learning

Eyeball method: all lines go through *(0,0)*.



- the function $y = 0 + c \cdot x$

intercept coefficient

- A linear model

Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

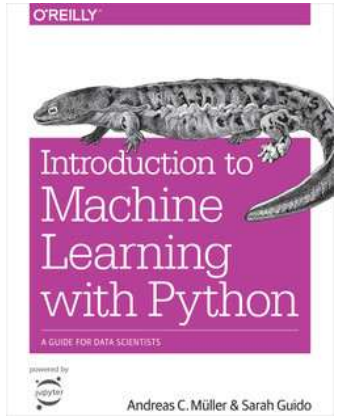
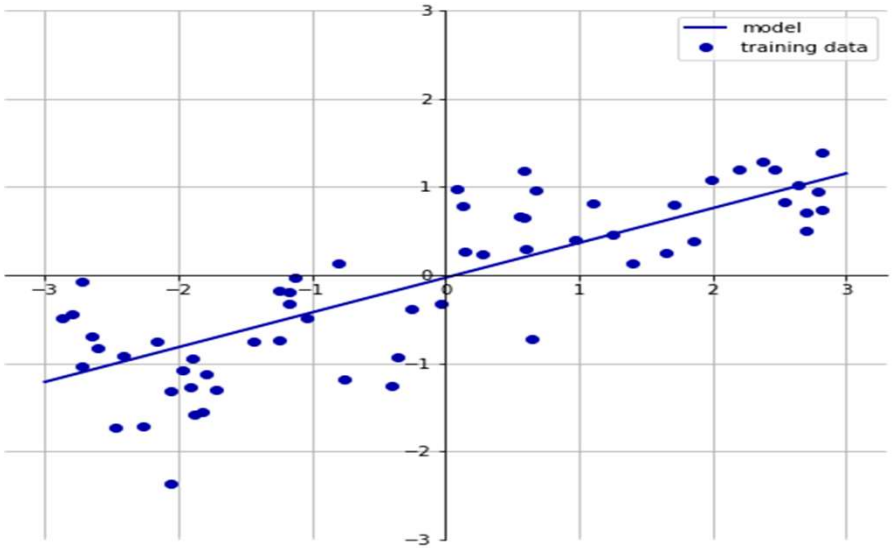


Figure 2-3

Introduction to Machine Learning

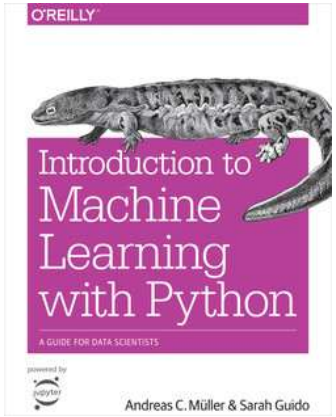
We need the fit() function of scikit-learn

.



A linear model $y = 0 + C \bullet X$

intercept coefficient

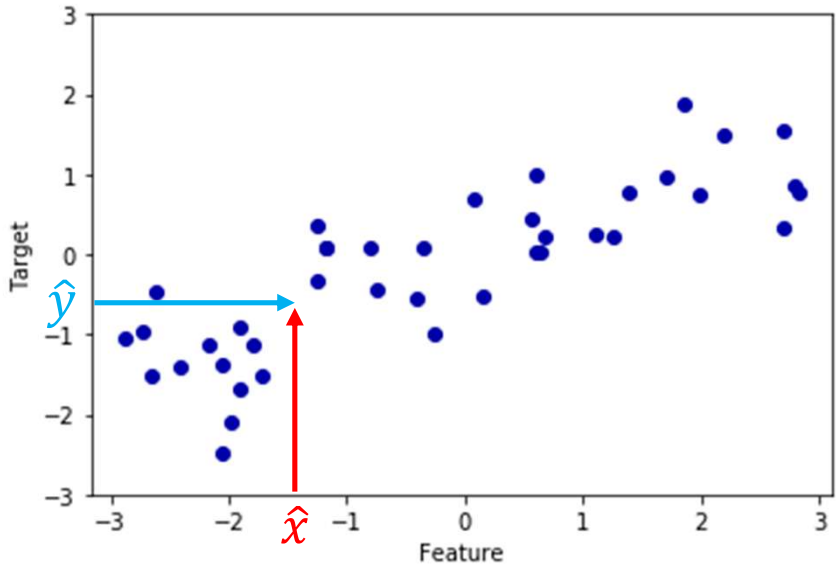


Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Page 49

Introduction to Machine Learning

Predict for this dataset the value at $\hat{x} = -1.5$.



- We fit the function between input X and output y

```
lr = LinearRegression().fit(X_train, y_train)
```

```
lr.predict([[-1.5]])
```

```
array([-0.623])
```

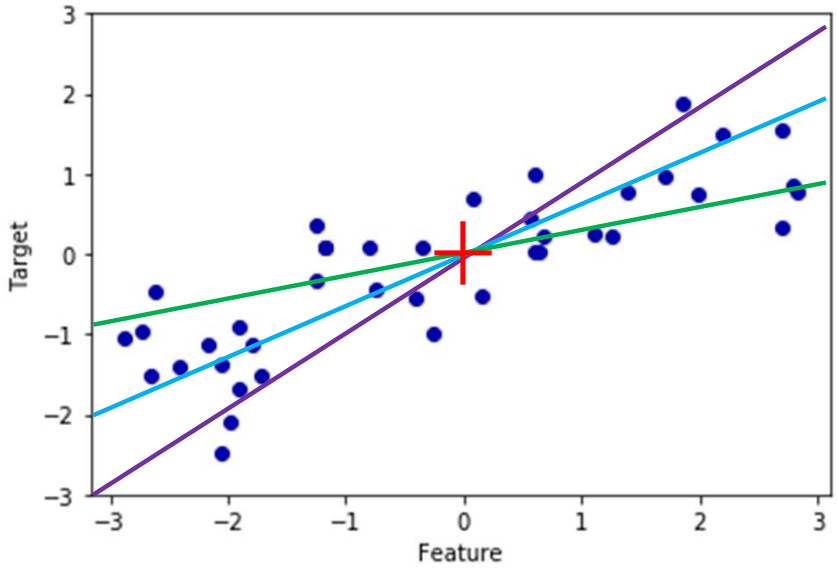
A linear model $-0.623 = -0.032 + -0.591$

Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

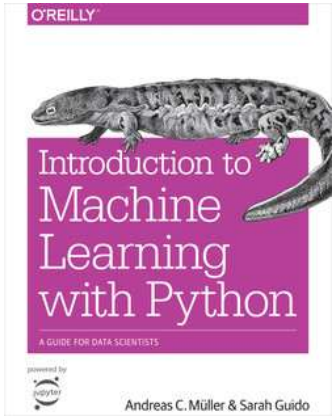
Figure 2-3

Introduction to Machine Learning

Fitting a lines go through points *y*




```
lr = LinearRegression().fit(X_train, y_train)
```

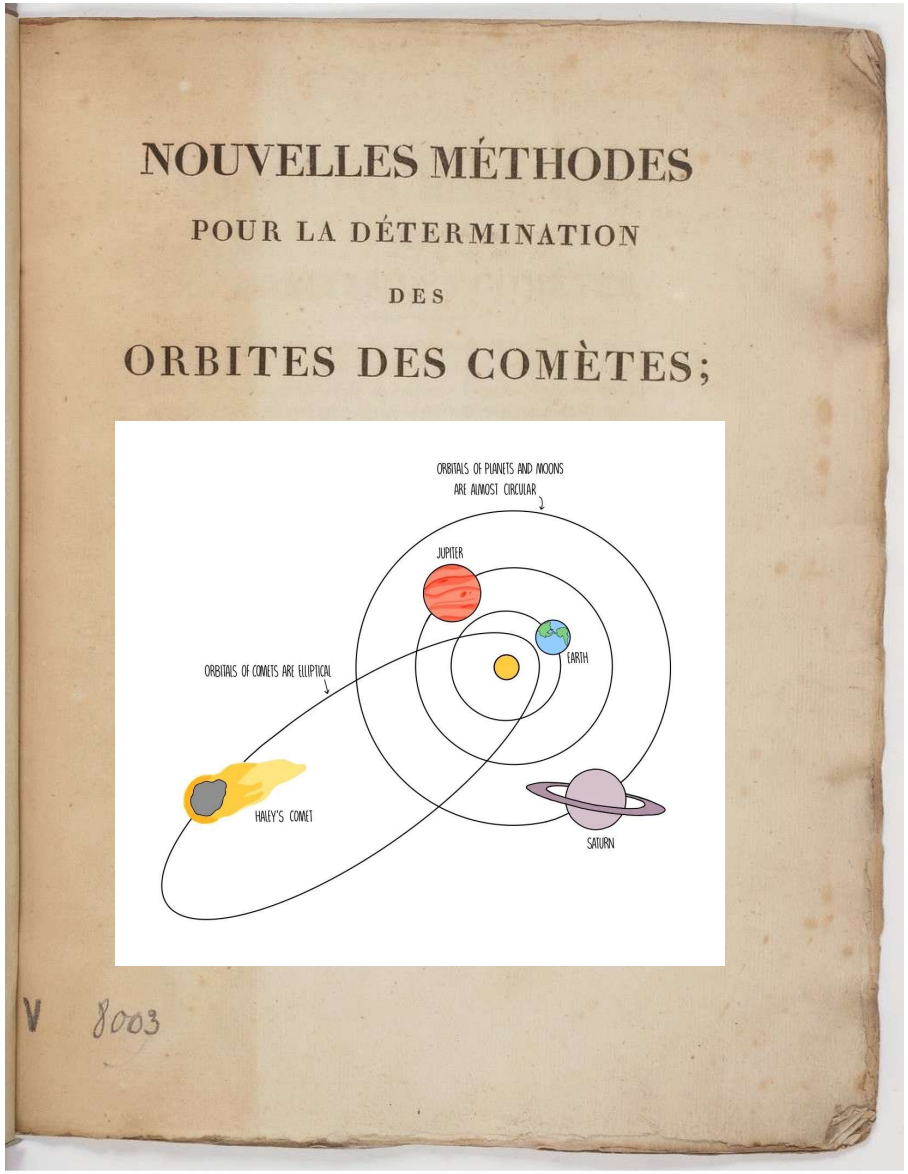


Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Page 49

Introduction to Machine Learning

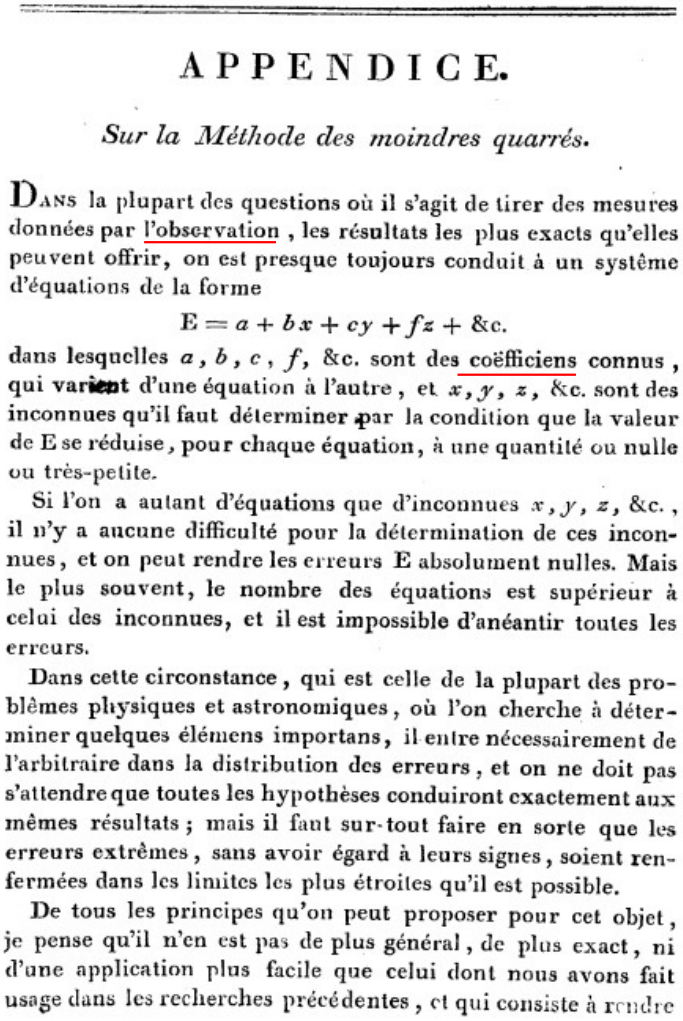
Origin of `fit(X_train, y_train)` : “la Méthode des moindres carrés”



Source gallica.bnf.fr / Bibliothèque nationale de France

Introduction to Machine Learning

Origin of `fit(X_train, y_train)` : “la Méthode des moindres carrés”



APPENDICE.

Sur la Méthode des moindres quarrés.

DANS la plupart des questions où il s'agit de tirer des mesures données par l'observation, les résultats les plus exacts qu'elles peuvent offrir, on est presque toujours conduit à un système d'équations de la forme

$$E = a + bx + cy + fz + \&c.$$

dans lesquelles $a, b, c, f,$ &c. sont des coëfficiens connus, qui varient d'une équation à l'autre, et $x, y, z,$ &c. sont des inconnues qu'il faut déterminer par la condition que la valeur de E se réduise, pour chaque équation, à une quantité ou nulle ou très-petite.

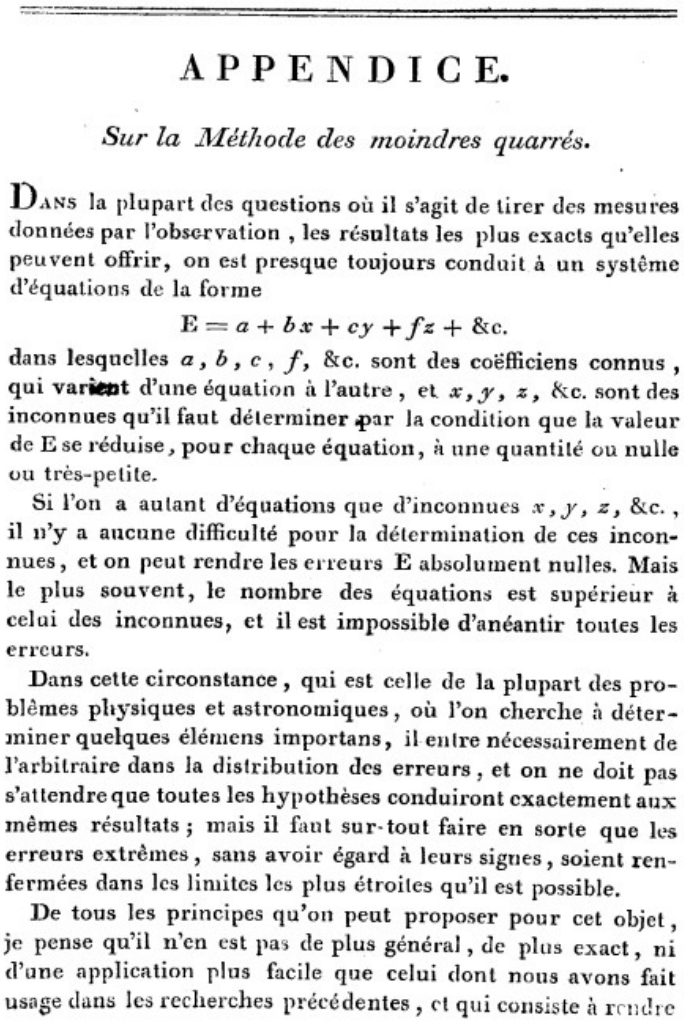
Si l'on a autant d'équations que d'inconnues $x, y, z,$ &c., il n'y a aucune difficulté pour la détermination de ces inconnues, et on peut rendre les erreurs E absolument nulles. Mais le plus souvent, le nombre des équations est supérieur à celui des inconnues, et il est impossible d'anéantir toutes les erreurs.

Dans cette circonstance, qui est celle de la plupart des problèmes physiques et astronomiques, où l'on cherche à déterminer quelques élémens importans, il entre nécessairement de l'arbitraire dans la distribution des erreurs, et on ne doit pas s'attendre que toutes les hypothèses conduiront exactement aux mêmes résultats; mais il faut sur-tout faire en sorte que les erreurs extrêmes, sans avoir égard à leurs signes, soient renfermées dans les limites les plus étroites qu'il est possible.

De tous les principes qu'on peut proposer pour cet objet, je pense qu'il n'en est pas de plus général, de plus exact, ni d'une application plus facile que celui dont nous avons fait usage dans les recherches précédentes, et qui consiste à rendre

Introduction to Machine Learning

Origin of `fit(X_train, y_train)` : “least square method”



APPENDICE.

Sur la Méthode des moindres quarrés.

DANS la plupart des questions où il s'agit de tirer des mesures données par l'observation, les résultats les plus exacts qu'elles peuvent offrir, on est presque toujours conduit à un système d'équations de la forme

$$E = a + bx + cy + fz + \&c.$$

dans lesquelles $a, b, c, f,$ &c. sont des coëfficiens connus, qui varient d'une équation à l'autre, et $x, y, z,$ &c. sont des inconnues qu'il faut déterminer par la condition que la valeur de E se réduise, pour chaque équation, à une quantité ou nulle ou très-petite.

Si l'on a autant d'équations que d'inconnues $x, y, z,$ &c., il n'y a aucune difficulté pour la détermination de ces inconnues, et on peut rendre les erreurs E absolument nulles. Mais le plus souvent, le nombre des équations est supérieur à celui des inconnues, et il est impossible d'anéantir toutes les erreurs.

Dans cette circonstance, qui est celle de la plupart des problèmes physiques et astronomiques, où l'on cherche à déterminer quelques élémens importans, il entre nécessairement de l'arbitraire dans la distribution des erreurs, et on ne doit pas s'attendre que toutes les hypothèses conduiront exactement aux mêmes résultats; mais il faut sur-tout faire en sorte que les erreurs extrêmes, sans avoir égard à leurs signes, soient renfermées dans les limites les plus étroites qu'il est possible.

De tous les principes qu'on peut proposer pour cet objet, je pense qu'il n'en est pas de plus général, de plus exact, ni d'une application plus facile que celui dont nous avons fait usage dans les recherches précédentes, et qui consiste à rendre

Simple linear regression

linear model

$$y = \alpha + \beta x$$

Several points with error ε_i

$$y_i = \alpha + \beta x_i + \varepsilon_i .$$

Error ε_i should be as small as possible

$$\hat{\varepsilon}_i = y_i - \alpha - \beta x_i .$$

Minimize the sum of all errors ε_i

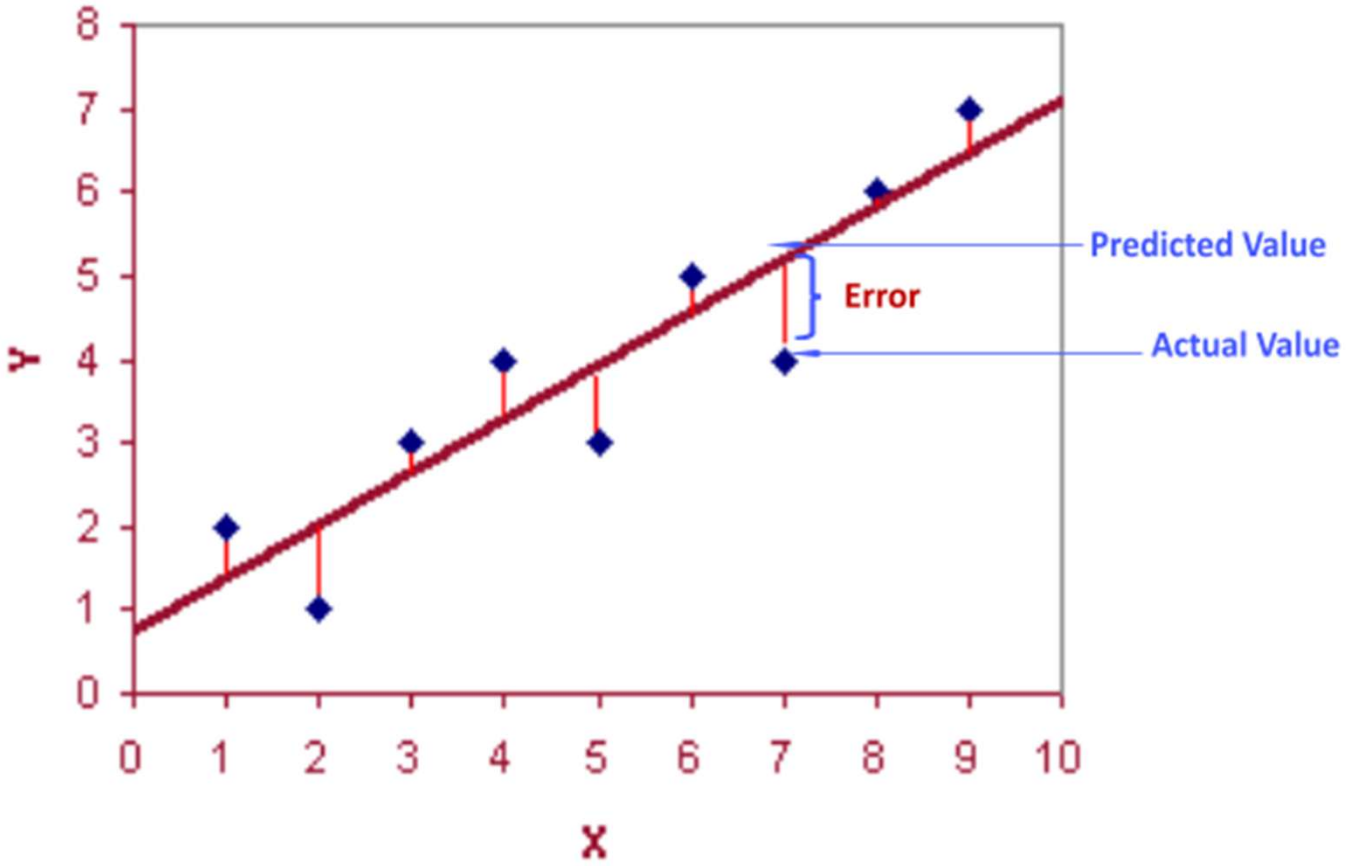
$$\sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 .$$

$$\sum_{i=1}^{N} \left(\hat{y}_i - y_i \right)^2$$

```
sum( (actual_y[i]- predicted_y[i])**2  for i in range(N) )
```

Least Square method

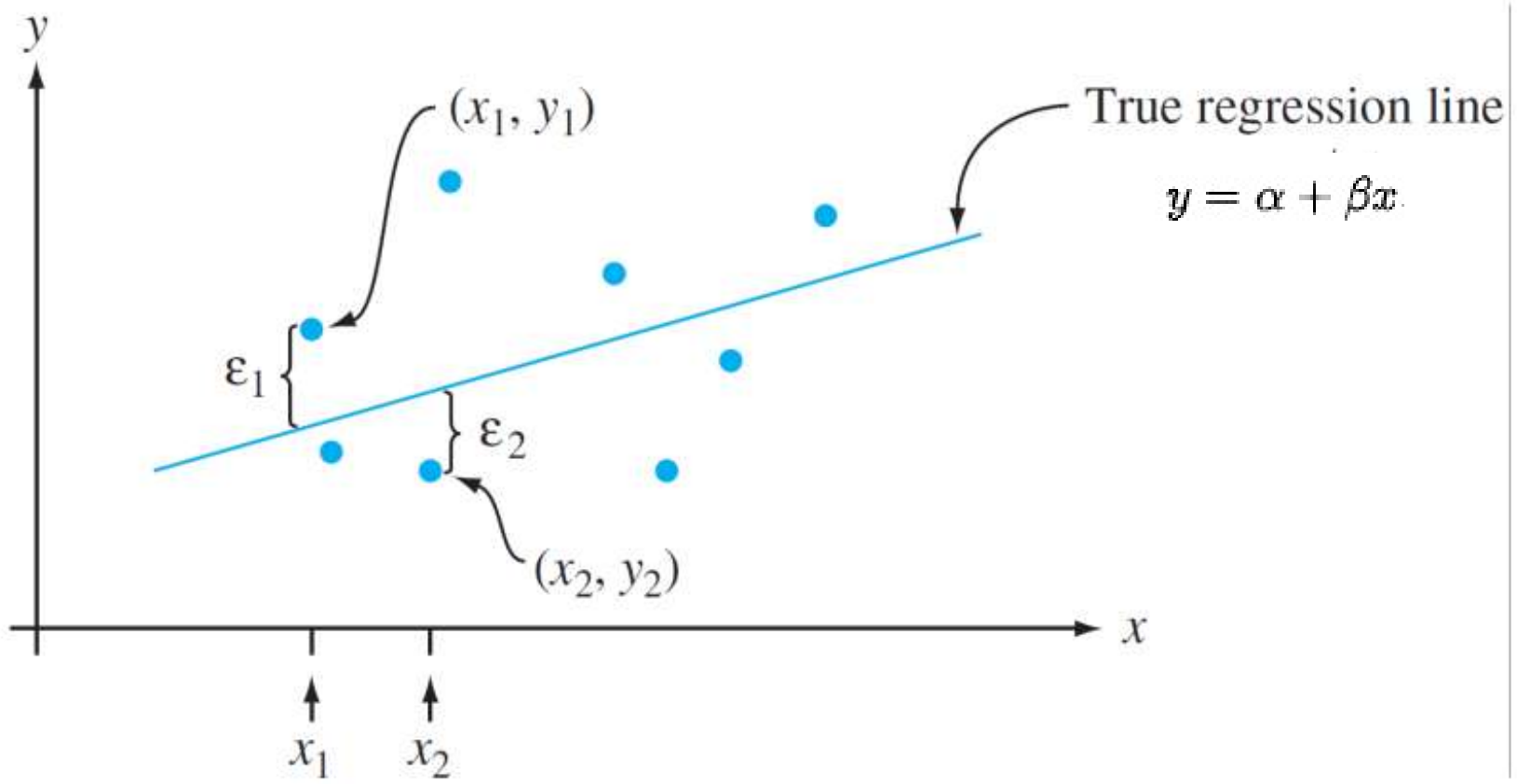
- You are given two arrays: `actual_y` and `predicted_y`



- Minimize the Error

Linear regression

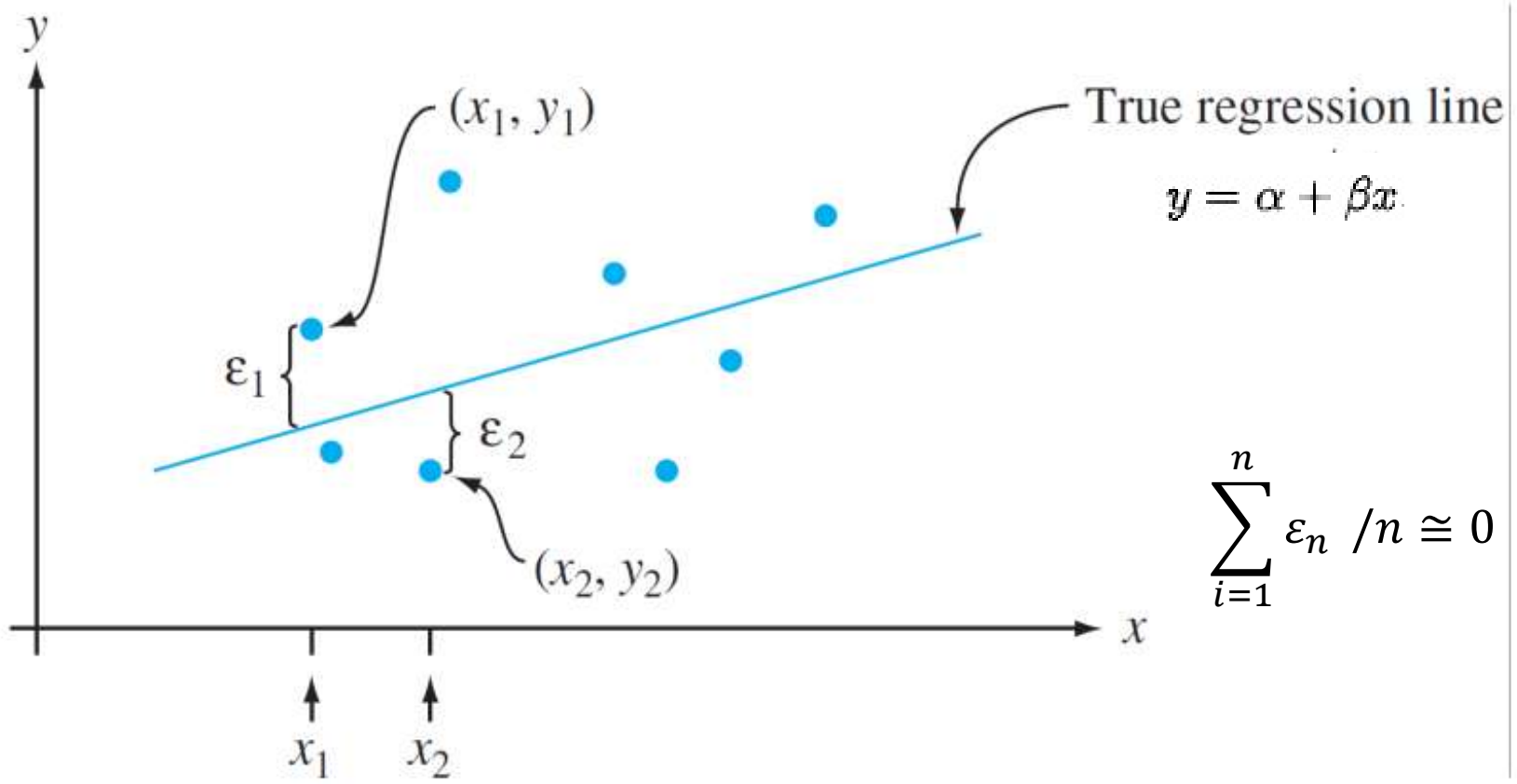
- The points (x_1, y_1), …, (x_n, y_n) resulting from n independent observations will then be scattered about the true regression line:



Yolanda Hagar, Simple Linear Regression, University of Colorado, 2016

Linear regression

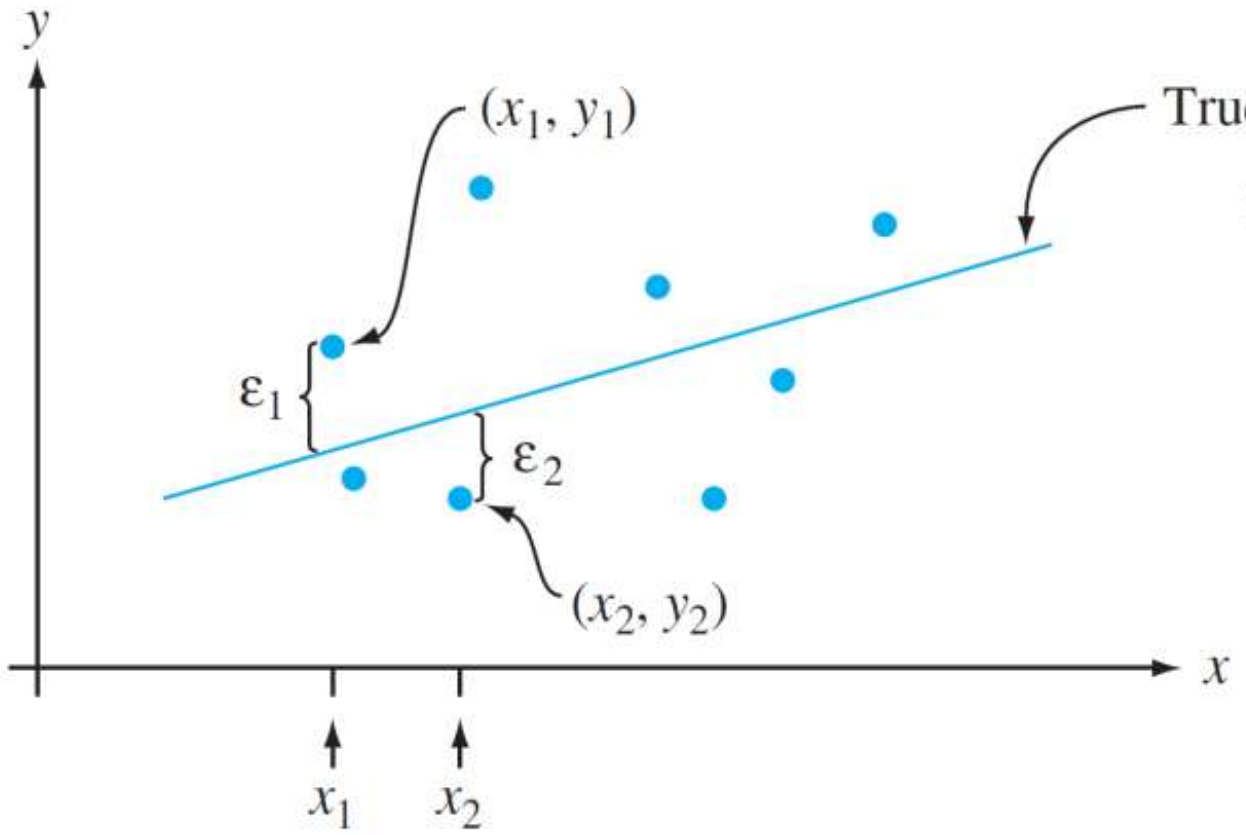
- The points $(x_1, y_1), \ldots, (x_n, y_n)$ resulting from n independent observations will then be scattered about the true regression line:



Yolanda Hagar, Simple Linear Regression, University of Colorado, 2016

Linear regression

- The points $(x_1, y_1), \ldots, (x_n, y_n)$ resulting from n independent observations will then be scattered about the true regression line:



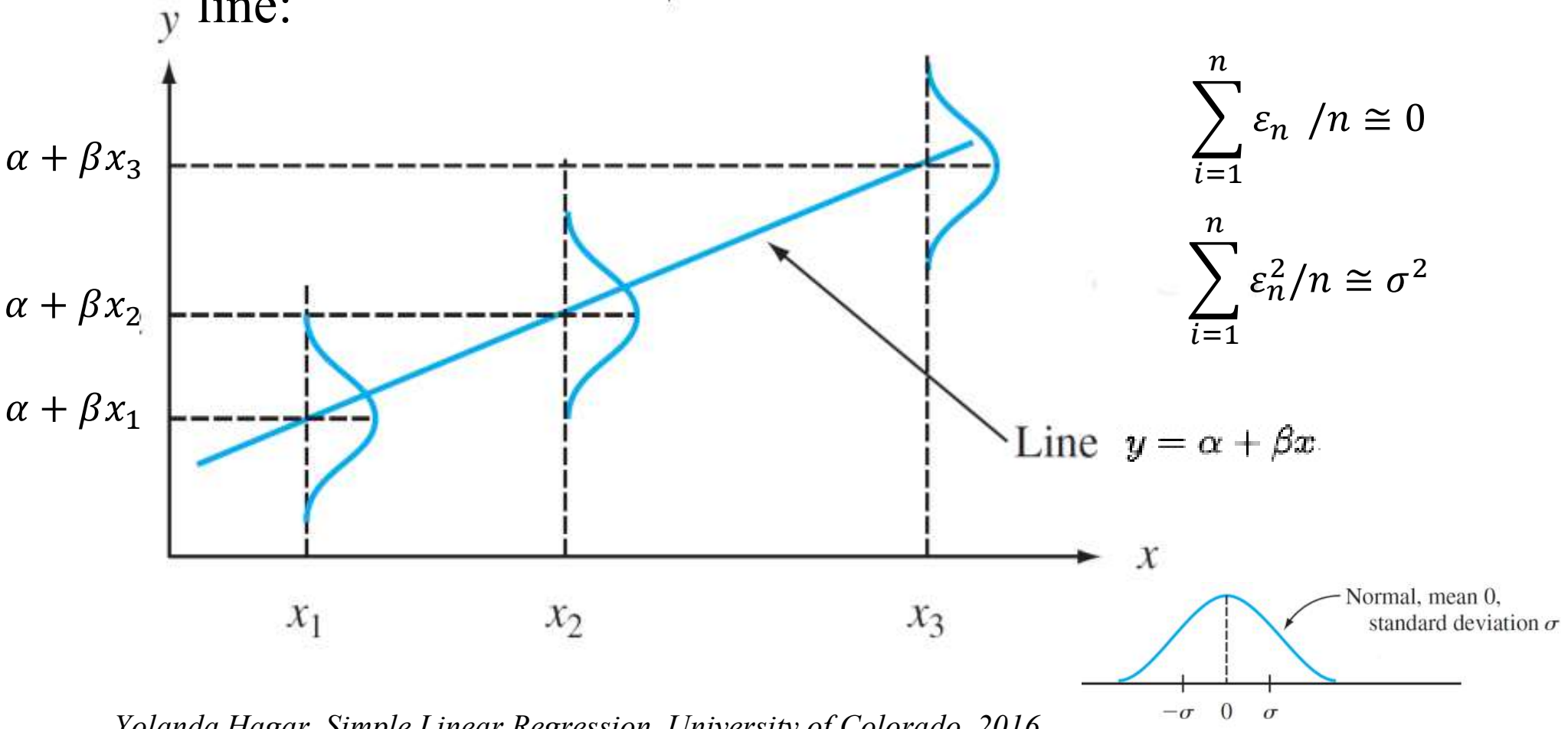
$$\sum_{i=1}^{n} \varepsilon_n^2/n \cong \sigma^2$$

$$\sum_{i=1}^{n} \varepsilon_n^2/n \cong 0$$

Yolanda Hagar, Simple Linear Regression, University of Colorado, 2016

Linear regression

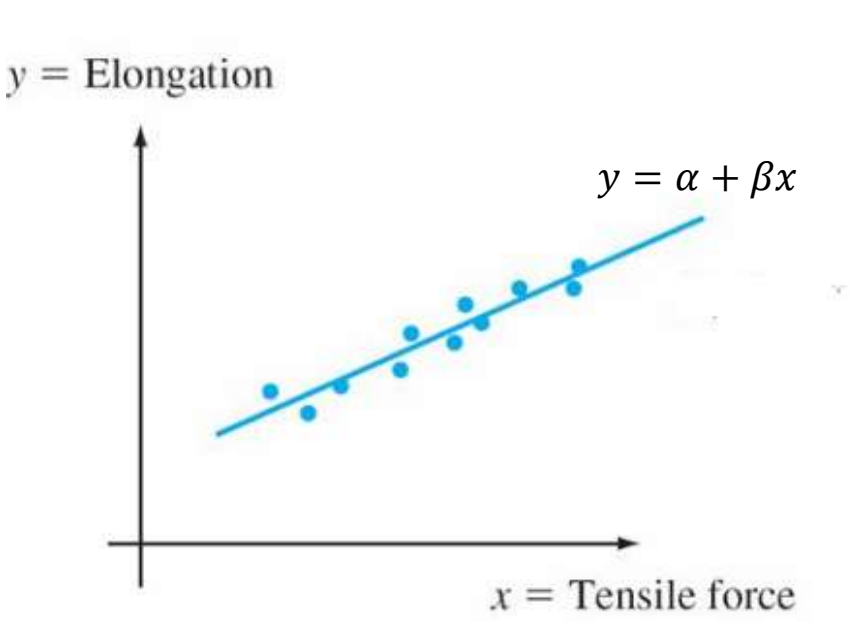
- The points (x_1, y_1), …, (x_n, y_n) resulting from n independent observations will then be scattered about the true regression line:

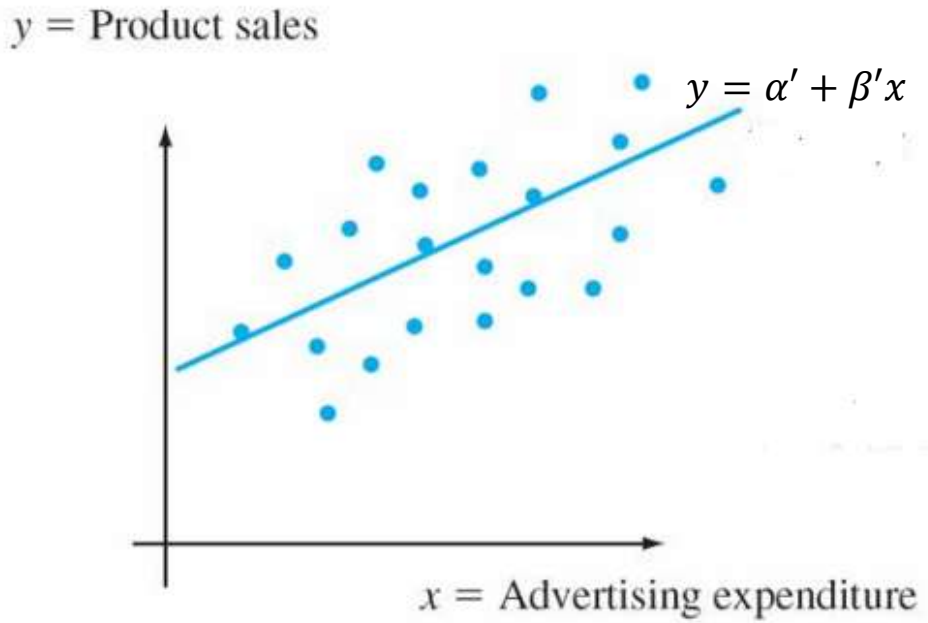


Yolanda Hagar, Simple Linear Regression, University of Colorado, 2016

Linear regression

- The parameter σ^2 determines the amount of spread about the true regression line. Two separate examples:





Yolanda Hagar, Simple Linear Regression, University of Colorado, 2016

The best fit

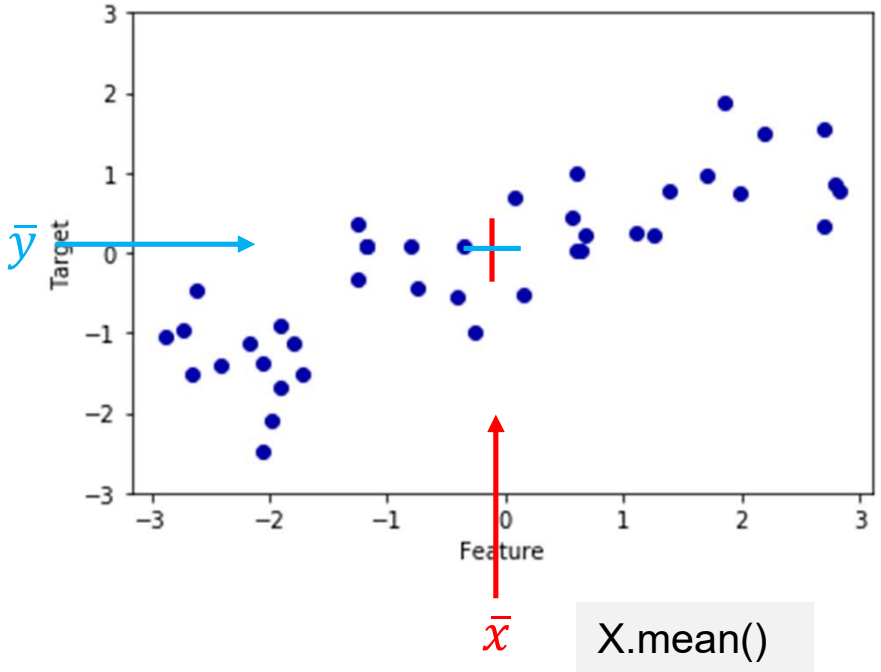
Minimize the sum of all errors

$$\sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 \,. \qquad (\hat{\alpha}, \hat{\beta}) = argmin(\sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2)$$

Two equations, two unknowns: $(\hat{\alpha}, \hat{\beta})$

When $\hat{\beta}$ is known, $\hat{\alpha}$ is easy to calculate:

y.mean()



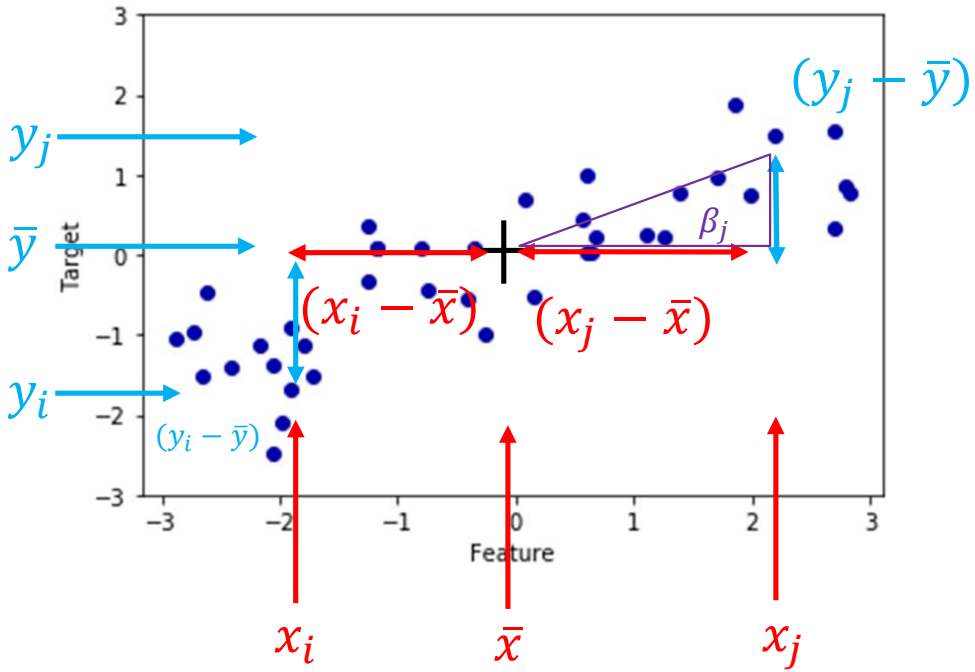
$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

~0.0 ~0.0 ~0.0

The best fit

Two equations, two unknowns: $(\hat{\alpha}, \hat{\beta})$

$\hat{\beta}$ is a bit harder to calculate, it is the weigthed average of all slopes: $\frac{(y_i - \bar{y})}{(x_i - \bar{x})}$



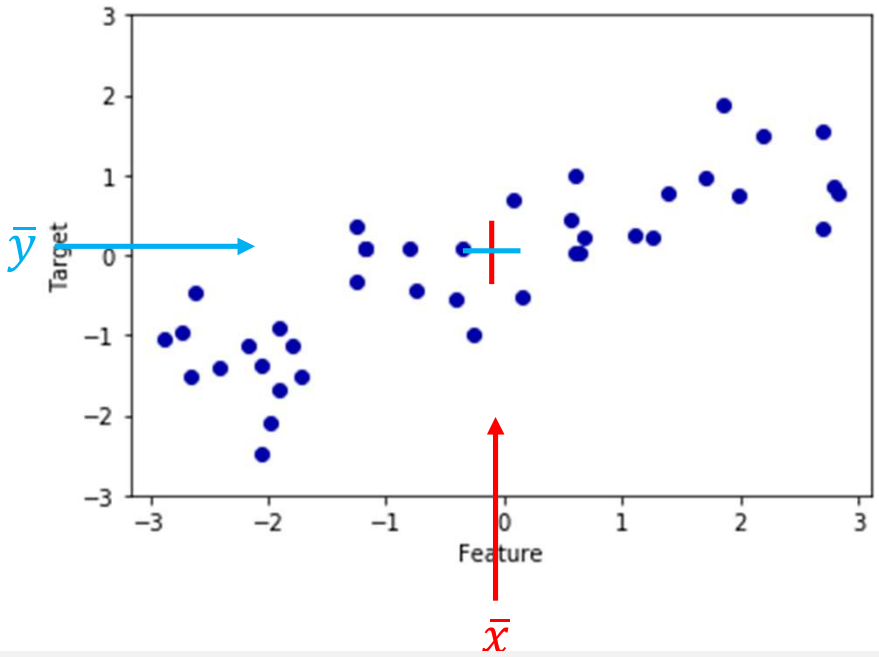
$$\hat{\beta} = \sum_{i=1}^{n} w_i \frac{(y_i - \bar{y})}{(x_i - \bar{x})}$$

$$w_i = \frac{(x_i - \bar{x})^2}{\sum_{j=1}^{n} (x_j - \bar{x})^2}$$

np.sum() X.mean() y.mean()

The best fit

When $\hat{\beta}$ is known, $\hat{\alpha}$ is easy to calculate:



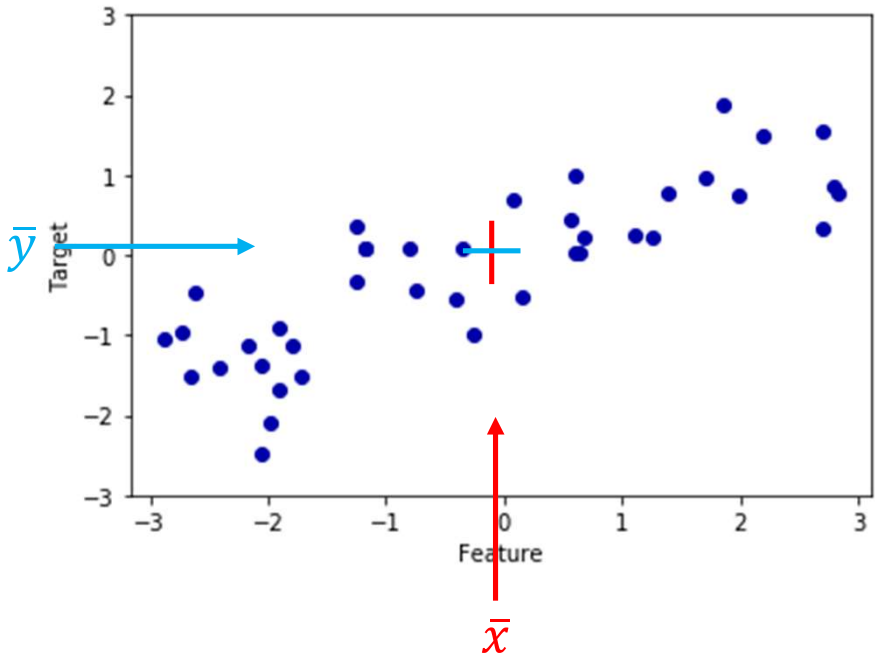
$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

```
from sklearn.linear_model import LinearRegression
X, y = mglearn.datasets.make_wave(n_samples=60)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)

alpha = y_train.mean() - model.coef_ * X_train.mean()
print("Our estimate of alpha is %.4f versus the LR model.intercept %.4f" % (alpha, model.intercept_))
```

Our estimate of alpha is -0.0939 versus the LR model.intercept -0.0939

The best fit

$\hat{\beta}$ is a bit harder to calculate, it is the weigthed average of all slopes: $\frac{(y_i-\bar{y})}{(x_i-\bar{x})}$



$$\hat{\beta} = \sum_{i=1}^{n} w_i \frac{(y_i - \bar{y})}{(x_i - \bar{x})}$$

```
dy = y_train - y_train.mean()
dX = (X_train - X_train.mean()).reshape((-1,))

wX = (dX ** 2) / ((dX ** 2).sum())
beta = (wX * (dy / dX)).sum()
print("Our estimate of beta is %.4f versus the LR model.coef %.4f" % (beta, model.coef_))
```

Our estimate of beta is 0.5242 versus the LR model.coef 0.5242

Linear models for Regression

For a single feature the model is:

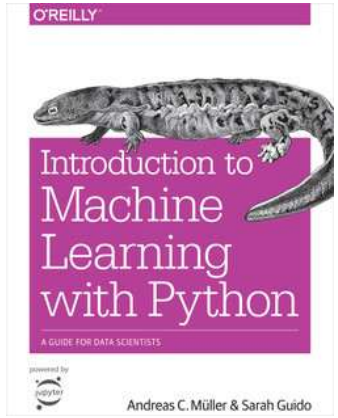
$$\hat{y} = w[0] * x[0] + b$$

For multiple features the model is:

$$\hat{y} = \beta_1 * x[0] + \beta_2 * x[1] + \cdots + \beta_p * x[p] + \beta_0$$

In matrix-notation the multiple-features model is:

$$\hat{y} = W * X + b$$



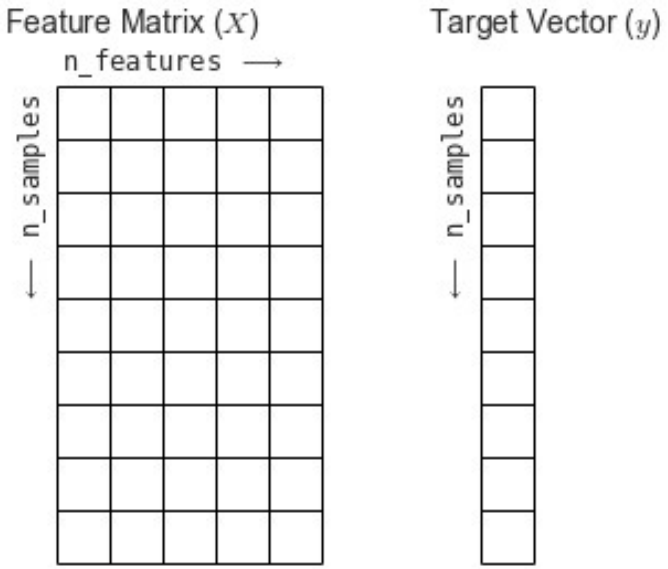
Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Page 497

Linear models for Regression

$$\hat{y} = W * X + b$$

Remember Scikit-learn's data format:



Jake VanderPlas, Python Data Science Handbook, O'Reilly Media, November 2016

Page 346

Real world example

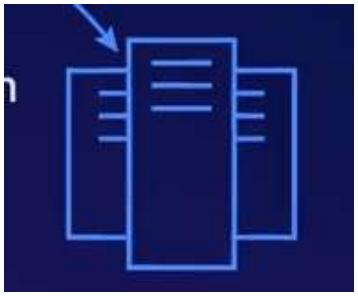
What the weight of a kid?

Expertise

- Depends on the organization
- Interpret the information in the right way

Evaluate baseline model – version 0.0

A child is half the weight of an adult



Dataset

Expertise

- Depends on the organization
- Interpret the information in the right way

Evaluate baseline model – version 0.0

regensburg_pediatric_appendicitis

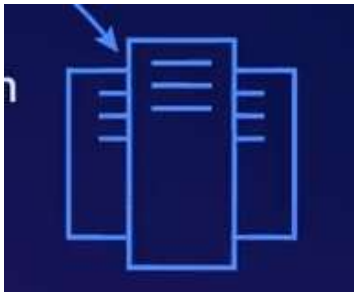
56 Features

	Feature Name	Type	Distinct/Missing Values
?	Management (target)	string	4 distinct values 1 missing attributes
▬	Age	numeric	577 distinct values 1 missing attributes
▬	BMI	numeric	510 distinct values 27 missing attributes
?	Sex	string	2 distinct values 2 missing attributes
▬	Height	numeric	187 distinct values 26 missing attributes
▬	Weight	numeric	268 distinct values 3 missing attributes

Dataset

Evaluate baseline model – version 0.0

A child is half the weight of an adult



regensburg_pediatric_appendicitis

Dataset

43.17 kg

```
import numpy as np
from sklearn.datasets import fetch_openml

patients = fetch_openml(name='regensburg_pediatric_appendicitis', version=1)
w = np.array(patients.data.Weight))
w = w[~np.isnan(h)]

print("Average weight of a child at Regensburg hospital is %.2f kg" % w.mean())
```

Evaluate Baseline model – version 0.0

A child is half the weight of an adult

regensburg_pediatric_appendicitis

Model

40 kg

Dataset

43.17 kg

- Only 8% off
- Our model is based on “prejudice”
- It does not use any feature of the child

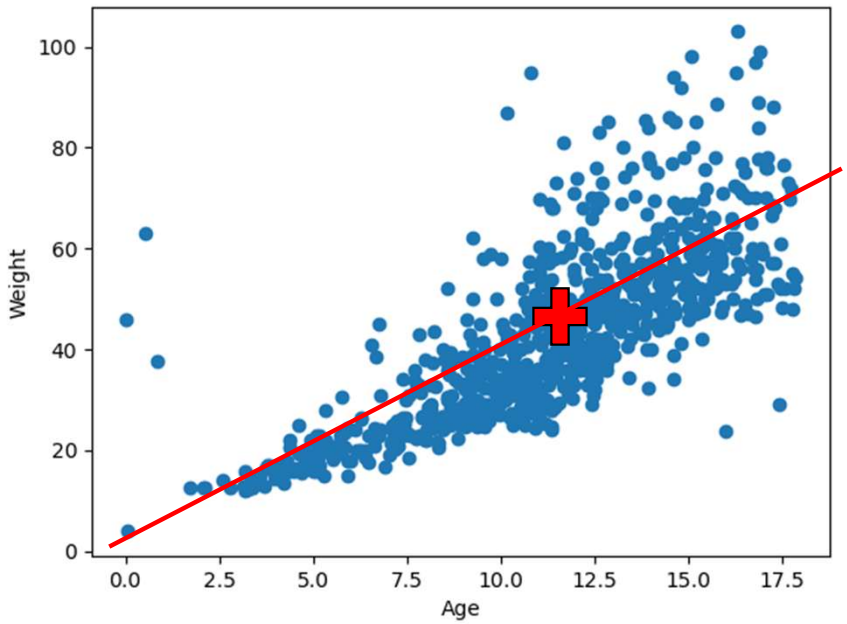
Germany	85.9 kg (189.4 lb)	69.2 kg (152.6 lb)	18+	Self-reported	2021

Evaluate informed models – version 0.3

- What happens when we add features of the child

regensburg_pediatric_appendicitis

Age



Dataset

43.17 kg

- Age is continues value

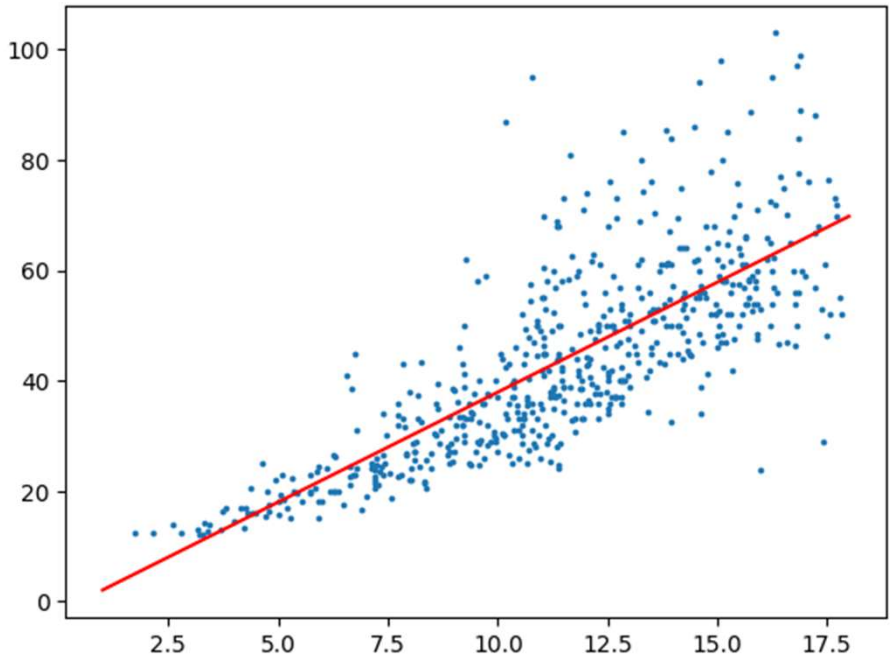
Regression model

Evaluate informed models – version 0.3

- So, $weight = BirthWeight + KiloPerYear * Age$

regensburg_pediatric_appendicitis

Dataset



Linear Regression model

$BirthWeight$

-1.91 kg

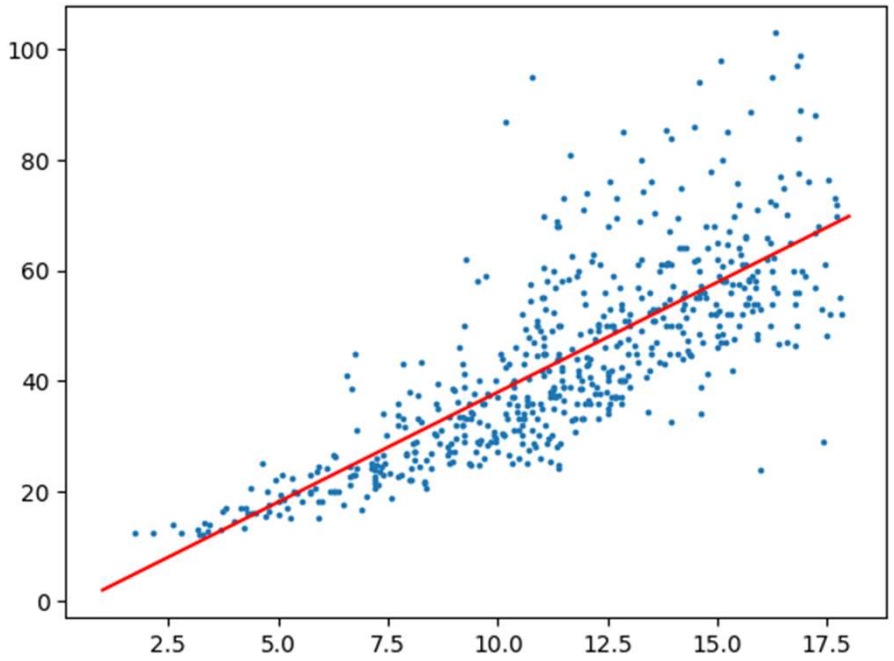
$KiloPerYear$

3.98 kg

Evaluate informed models – version 0.3

- So, $weight = BirthWeight + KiloPerYear * Age$

Linear Regression model



regensburg_pediatric_appendicitis

Training Dataset

$AverageAge$

11½

$AverageWeight$

43.65 kg

$AverageWeight$ at $AverageAge$

43.65 kg

Evaluate informed models – version 0.3

- So, $weight = BirthWeight + KiloPerYear * Age$

Data preparation

regensburg_pediatric_appendicitis

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_openml
import numpy as np

patients = fetch_openml(name='regensburg_pediatric_appendicitis', version=1)
a = np.array(patients.data.Age[patients.data.Age>1.0])
w = np.array(patients.data.Weight[patients.data.Age>1.0])

a = a[~np.isnan(w)]
w = w[~np.isnan(w)]

A = a[:, np.newaxis]

X_train, X_test, y_train, y_test = train_test_split(A, w, random_state=42)
```

Evaluate informed models – version 0.3

- So, $weight = BirthWeight + KiloPerYear * Age$

Model fit

regensburg_pediatric_appendicitis

```
from sklearn.linear_model import LinearRegression

model = LinearRegression(fit_intercept=True)
model.fit(X_train, y_train)

print("Average weight-gain of a child at Regensburg hospital is %.2f kg per year" % model.coef_)
```

```
Average weight-gain of a child at Regensburg hospital is 3.98 kg per year
```

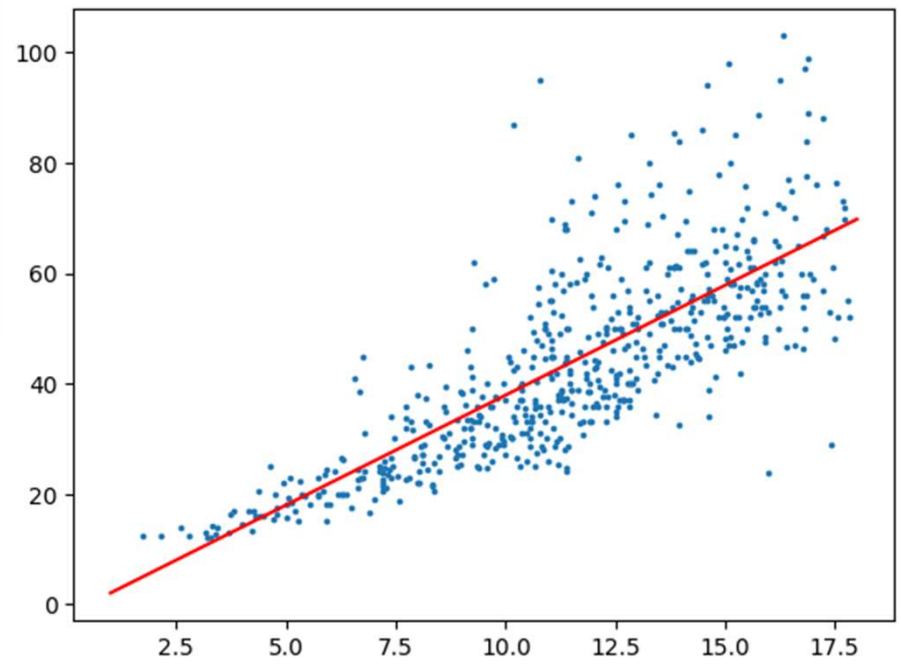
Evaluate informed models – version 0.3

- So, $weight = BirthWeight + KiloPerYear * Age$

Model check

regensburg_pediatric_appendicitis

```
import matplotlib.pyplot as plt

xfit = np.linspace(1,18)

Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)

plt.scatter(X_train, y_train, s=3)
plt.plot(xfit, yfit,c='r');
```



R2 score: 0.60 ± 0.06

Evaluate informed models – version 0.3 outliers

- So, $weight = BirthWeight + KiloPerYear * Age$

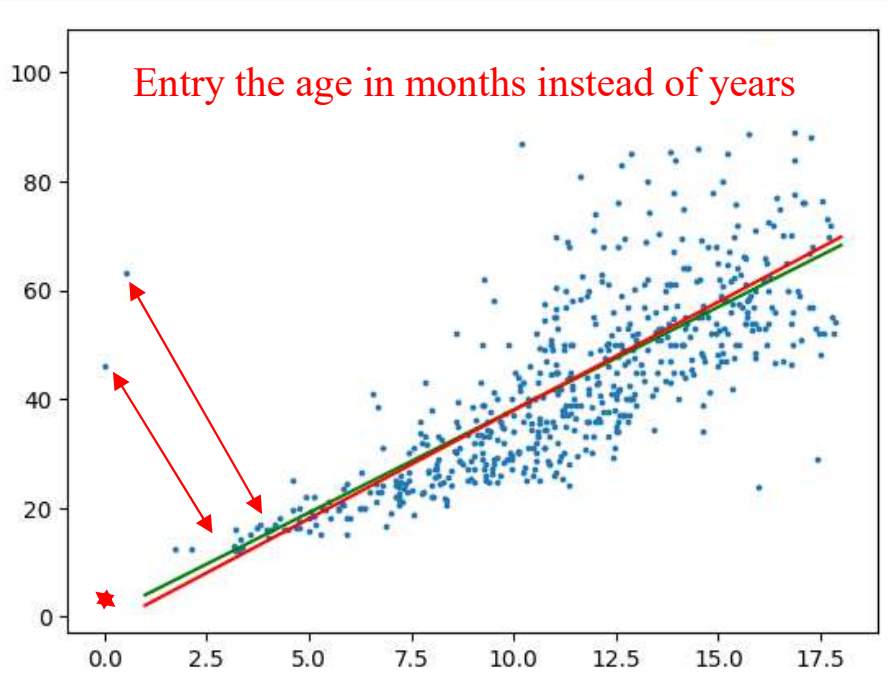
-1.91 kg → +0.20 kg 3.95 kg / y → 3.78 kg / y Δ 4%

Model check

regensburg_pediatric_appendicitis

```
import matplotlib.pyplot as plt

xfit = np.linspace(1,18)

Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)

plt.scatter(X_train, y_train, s=3)
plt.plot(xfit, yfit,c='r');
```

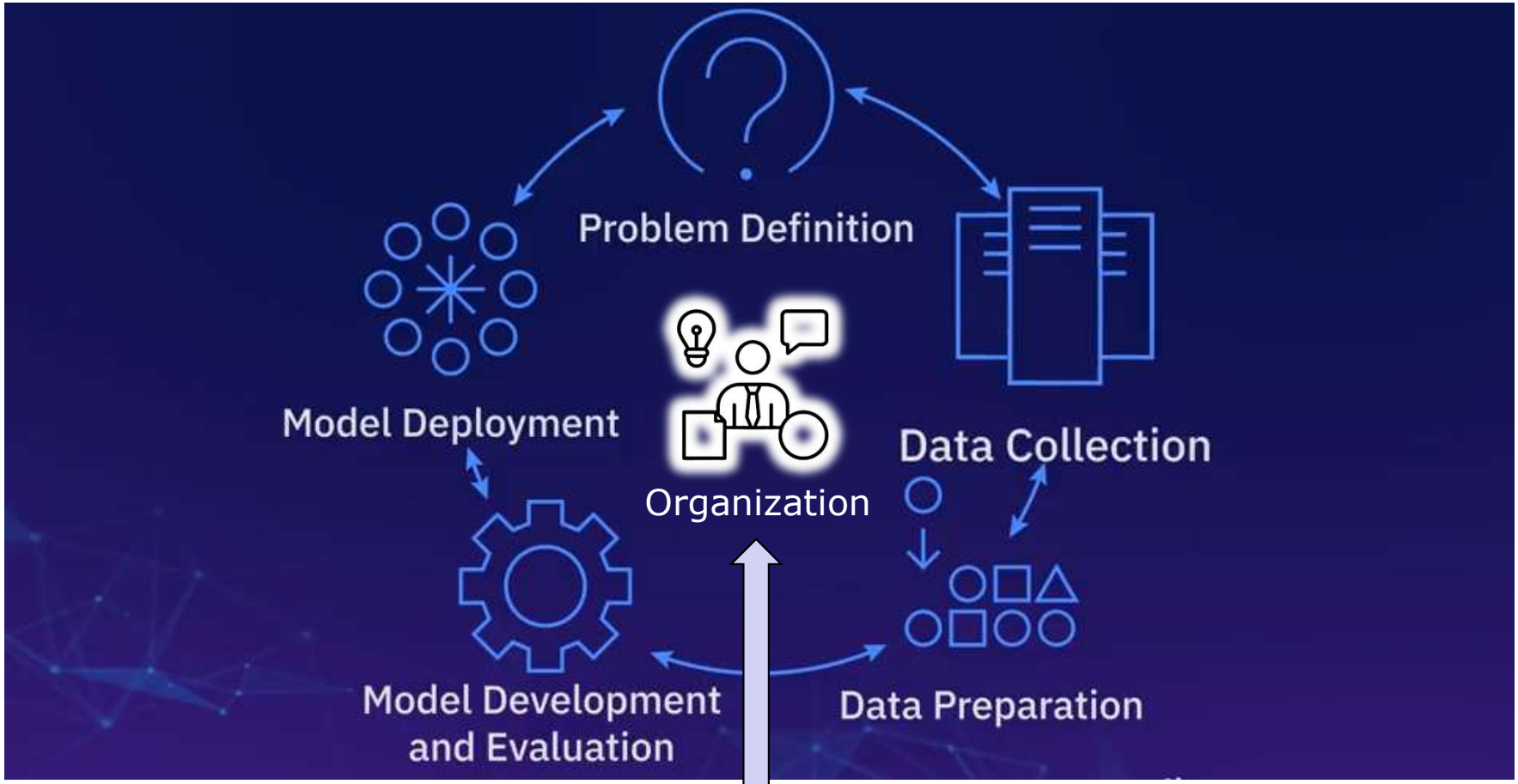
3 extra training points



R2 score: 0.57 ± 0.03

Introduction to Machine Learning

Building a predictive model is a circular process.

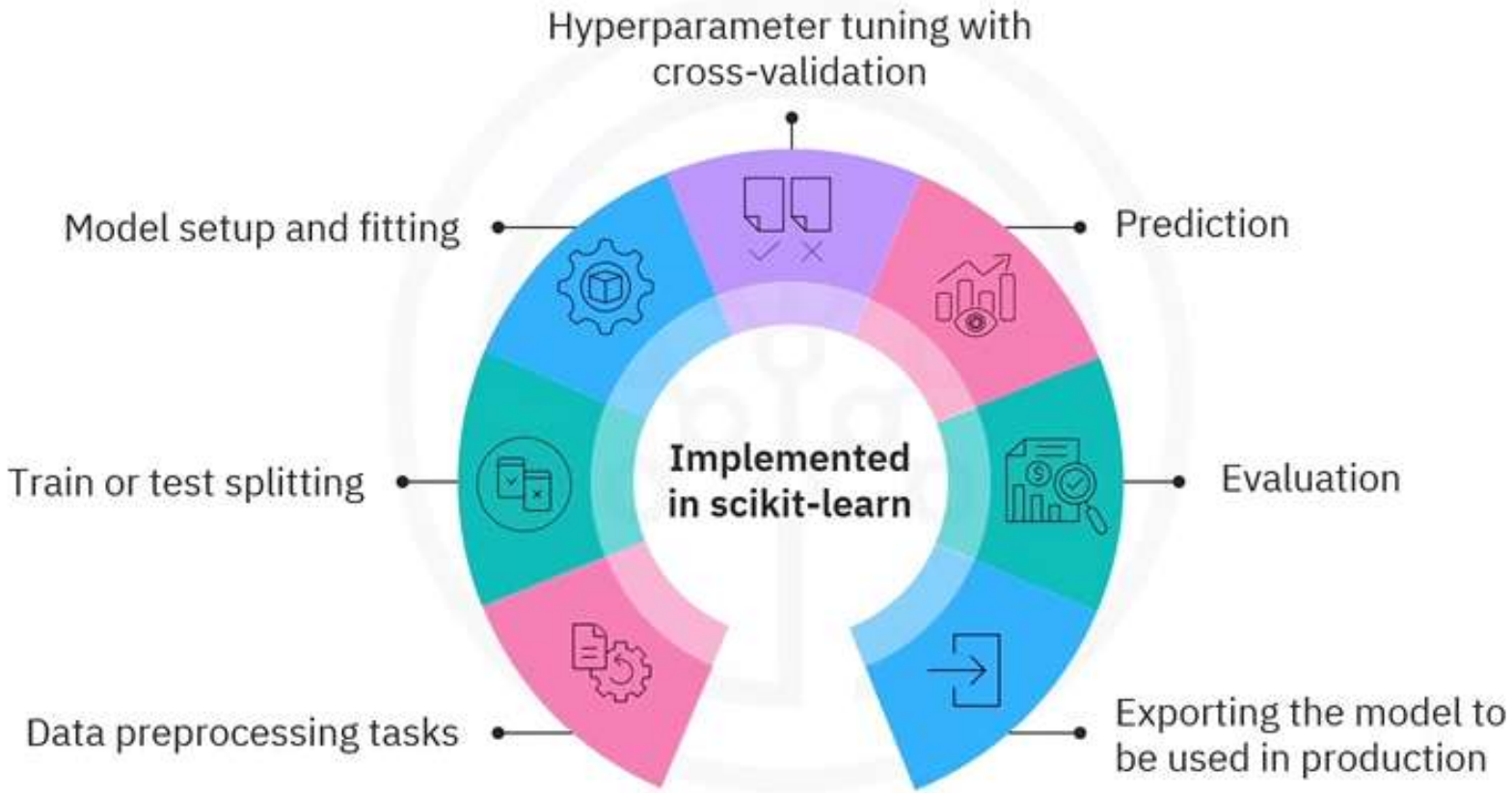


Expertise

- Depends on the organization
- Interpret the information in the right way

Introduction to Machine Learning

Building a predictive model is a circular process.

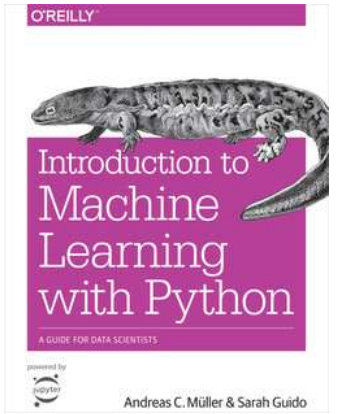


Joseph Santarcangelo & Joss Grossman, Machine Learning with Python, IBM Skills network, 2025

Introduction to Machine Learning

- 2.3.3 Linear Regression Models

- Strengths:
  - Easy to understand
  - Fast to train, fast to predict
  - Scales well to large training sets, even when sparse
- Weakness:
  - On datasets with many features, the coefficients reflect direct & indirect correlations
  - On datasets with many features, the complexity has to be controlled with *regularization*



Andreas C. Müller, Sarah Guido, Introduction to Machine Learning with Python, O'Reilly Media, October 2016

Page 69

Conclusion

Learning outcomes of this course covered today

- What is a good fit of the relation between two features